A Comparative Evaluation of Implicit Authentication Schemes

Hassan Khan, Aaron Atwater, and Urs Hengartner

Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
{h37khan,aatwater,urs.hengartner}@uwaterloo.ca

Abstract. Implicit authentication (IA) schemes use behavioural biometrics to continuously and transparently authenticate mobile device users. Several IA schemes have been proposed by researchers which employ different behavioural features and provide reasonable detection accuracy. While these schemes work in principle, it is difficult to comprehend from these individual efforts which schemes work best (in terms of detection accuracy, detection delay and processing complexity) under different operating conditions (in terms of attack scenarios and availability of training and classification data). Furthermore, it is critical to evaluate these schemes on unbiased, real-world datasets to determine their efficacy in realistic operating conditions. In this paper, we evaluate six diverse IA schemes on four independently collected datasets from over 300 participants. We first evaluate these schemes in terms of: accuracy; training time and delay on real-world datasets; detection delay; processing and memory complexity for feature extraction, training and classification operations; vulnerability to mimicry attacks; and deployment issues on mobile platforms. We also leverage our real-world device usage traces to determine the proportion of time these schemes are able to afford protection to device owners. Based on our evaluations, we identify: 1) promising IA schemes with high detection accuracy, low performance overhead, and near real-time detection delays, 2) common pitfalls in contemporary IA evaluation methodology, and 3) open challenges for IA research. Finally, we provide an open source implementation of the IA schemes evaluated in this work that can be used for performance benchmarking by future IA research.

1 Introduction

Smartphones are strongly tied to their owners' identity and contain personal data. In order to protect this data from unauthorized access, smartphones are equipped with authentication mechanisms including PINs, pass-locks, and facial and fingerprint recognition systems. These authentication mechanisms provide traditional all-or-nothing access control. However, smartphone use is characterized by short and frequent sessions, and PIN entry for every short session is inconvenient for users [18]. Furthermore, the all-or-nothing access approach is

A. Stavrou et al. (Eds.): RAID 2014, LNCS 8688, pp. 255–275, 2014.

unsuitable for smartphones where 40% of frequently accessed apps do not contain personal data [18]. Due to these usability issues, 50% of smartphone owners do not configure a pass-lock on their devices [29]. In addition to usability issues, pass-locks have been subject to shoulder surfing attacks and operating system flaws [40]. The facial and fingerprint recognition systems on modern high-end devices have also been shown to be vulnerable [1,30]. These security and usability limitations of primary authentication mechanisms have prompted researchers to develop behaviour-based methods of recognizing and validating the identity of smartphone users. These behaviour-based authentication methods are known as implicit authentication (IA) schemes, which authenticate a user by using distinctive, measurable patterns of device use that are gathered from the device user without requiring deliberate actions [9]. IA schemes can be used as a secondary line of defense in multiple scenarios. For example, they might be used by enterprise or banking apps to ensure that a user's password has not been compromised. Alternatively, they provide a middle ground for the smartphone owners who do not employ pass-locks on their devices due to usability issues.

To provide IA support on smartphones, a variety of behaviour-based classifiers have been proposed [8,10,11,12,13,15,24,26,34,35,42]. Many of these behavioural classifiers have reasonably high accuracy rates, low performance overhead and reasonable detection delay. While these results appear to stand on their own, it is often difficult to compare different proposals. For example, some IA schemes based on touchscreen input behaviour [11,42] provide exceptional accuracies when they are evaluated on datasets collected by those individual efforts. However, Feng et al. [12] showed that on data collected in an uncontrolled environment, the accuracy of these approaches reduces significantly. Similarly, due to the unavailability of real-world datasets, it is not possible for these individual research efforts to accurately report the training and detection delay in an uncontrolled environment. Finally, a majority of existing IA proposals fall short of providing performance benchmarks (in terms of CPU and memory overhead) on smartphones. Consequently, it is difficult to understand the impact on user experience due to overhead on power-constrained smartphones by these schemes.

In addition to unreported performance numbers (in terms of detection delay and computational cost), many IA schemes use behavioural features, for which it is non-trivial to estimate the frequency or availability of such data. For example, characterizing a device owner's gait may be a useful discriminative tool for authentication, but is not useful if the device owner is stationary most of the time. Therefore, there is a need not only for datasets that allow IA schemes' evaluation in realistic scenarios, but an analysis of real-world behavioural patterns that may influence the appropriateness of deploying one scheme over another.

In this paper, we evaluate and compare six IA schemes using four independently collected datasets from multiple geographic locations, comprising over 300 participants. The objectives of this study are: 1) to quantify and compare the accuracies of these IA schemes on independently collected datasets from uncontrolled environments, 2) to use real-world traces to measure training and detection delays for these IA schemes, 3) to determine the performance overhead

of these IA schemes on mobile devices, 4) to determine the frequency of data availability for different behavioural features employed by these IA schemes, 5) to identify key research challenges in IA research, and 6) to release open source implementations of these IA schemes for performance benchmarking.

The IA schemes evaluated in this work use diverse behavioural biometrics including touchscreen input behaviour, keystroke patterns, gait patterns, call and text patterns, browser history, and location patterns. Some also combine touchscreen input behaviour with a device's micro-movements as a reaction to touch input and context information, respectively. Some of these IA schemes employ machine learning while others employ statistical measures for classification purposes. This diversity allows us to better scrutinize different aspects of these individual IA schemes and determine research challenges and best practices.

We evaluate these IA schemes on eight criteria: 1) accuracy, 2) data availability, 3) training delay, 4) detection delay, 5) CPU and memory overhead, 6) uniqueness of behavioural features, 7) vulnerability to mimicry attacks, and 8) deployment issues on mobile platforms. Our results show that while the majority of IA schemes provide reasonable accuracy with low detection delay, IA schemes based on touchscreen input behaviour outperform others by providing near real-time misuse detection with high accuracy. We find that some IA schemes perform well in terms of detection accuracy but frequently do not have enough data available for classification. We recommend choosing complementary sources of features to mitigate this problem and also to aid in preventing mimicry attacks. Finally, we release[1] our open source implementations of the six IA schemes evaluated in this paper.

2 Related Work and Background

In this section, we discuss related work and provide a brief description of the six IA schemes evaluated in this paper.

2.1 Related Work

Various IA schemes have been proposed as secondary authentication mechanisms to complement primary authentication mechanisms (such as PINs and passlocks). These schemes employ a variety of behavioural features including a user's location patterns [38], call/text patterns [35], keystroke patterns [8,13,26], proximity to known devices [21], gait patterns [14,17,27,28], and touchscreen input behaviour [10,11,12,15,24,34,42]. Furthermore, some authors have proposed combining behavioural features and contextual information from multiple sources [4,29,35]. While these research efforts demonstrate that these IA schemes and behavioural features work in principle, we provide a comparative evaluation of these schemes on independently collected datasets across a more comprehensive evaluation criteria than the original papers.

[1] `https://crysp.uwaterloo.ca/software/ia/`

To the best of our knowledge, a comparative evaluation of different IA schemes has not been performed yet. Serwadda et al. [33] perform a benchmark evaluation of three touch-based IA schemes using ten classification algorithms to evaluate which classifiers work best. While their analysis provides interesting insights, we aim to provide a comparative evaluation of IA schemes that employ different behavioural features. Furthermore, except [12] and [34], none of the other authors have provided a comparison with other schemes. We believe this is due to the effort required to implement another scheme and to collect data to perform empirical evaluations. Therefore, in addition to providing a comparative evaluation of IA schemes, by making the implementation and datasets publicly available, we will enable future researchers to quantify the efficacy of their approach with other related schemes. Finally, our findings will also provide better insights to researchers who are designing generic IA frameworks [7,9].

2.2 Implicit Authentication Schemes

For comparative evaluation, our goal is to compare IA schemes that rely on different behavioural features. To this end, we chose an IA scheme based on call/text/URL history and location [35], an IA scheme based on gait patterns [14], an IA scheme based on touch input behaviour [15], an IA scheme based on keystroke behaviour [13], an IA scheme based on touch and micro-movement behaviour [4], and an IA scheme based on touch behaviour and user context [12].

Before describing these IA schemes, we define some terms that are used throughout this paper. A *true accept* (TA) is when an access attempt by a legitimate user is granted; a *false reject* (FR) is when an access attempt by a legitimate user is rejected by the IA scheme. A *true reject* (TR) is when an access attempt by an adversary is rejected; a *false accept* (FA) is when an access attempt by an adversary is granted by the IA scheme. *Equal Error Rate* (EER) is the operating point where the rate of true accepts is equal to the rate of true rejects. In this work, *accuracy* is defined as $\frac{TA+TR}{TA+FA+TR+FR}$. We now provide a brief description of the IA schemes evaluated in this paper; interested readers are referred to the original papers for full descriptions of the respective methods.

Shi et al. IA Scheme (Shi-IA) [35]. Shi et al. [35] propose an IA scheme that uses good and bad events to determine an authentication score for a user. Good/habitual behaviour is determined by a phone call/text to a known number, a visit to a familiar website, and presence at habitual locations around a certain time-of-day. Similarly, bad behaviour is a phone call/text to an unknown number, a visit to an unfamiliar website, and presence at previously unseen locations. Passage of time since the last good event is also treated as a negative event and results in gradual decay of the authentication score. For empirical evaluations, the authors used data gathered from 50 participants, trained on 2 weeks of their usage data and evaluated on the remaining data. Their results indicate that 95% of the time an adversary can be detected using 16 or fewer usages of the devices with negligible false rejects (1 in 165). We choose Shi-IA for empirical evaluations because of the unique feature set it employs for IA.

Gait Pattern for User Identification (Gait-IA) [14]. Various authors have proposed using gait patterns for user identification on smartphones [14,17,27,28]. We chose Frank et al. [14] for empirical evaluations as the authors have made their implementation and dataset publicly available, making it easier to reproduce their results for verification purposes. Furthermore, they report significantly higher accuracy than other gait pattern based schemes. We also note that Frank et al. only propose employing gait patterns for user identification and not for IA; nevertheless, we evaluate whether gait behaviour can be used to effectively implicitly authenticate a user.

Frank et al. propose a time-delay embedding approach to gait recognition. Time-delay embedding is employed to reconstruct the state of an unknown dynamical system from observations of that system taken over time. The authors first extract features using time-delay embeddings and then perform noise-reduction over those features using principal component analysis (PCA) [20] on a short embedding of training data. PCA produces a projection from the time-delay embedding space to a lower dimension model space. These resulting features are then employed in an ANN classifier [2]. Empirical evaluation on walking data from 25 individuals (with the device in the front trouser pocket) resulted in 100% detection accuracy.

Touchalytics [15]. Touchscreen input behaviour has been widely investigated for IA [10,11,15,24,34,42]. For our empirical evaluations, we choose Touchalytics as the authors have made their implementation and dataset publicly available. Touchalytics, is an IA scheme that relies on the finger movement patterns of users on the touchscreen of their smartphone. Touchalytics uses data generated as a result of a user's normal interaction with the touchscreen and does not require him to perform special gestures. It operates by first recording the raw touch data and then extracting 31 features from the raw data. These features capture the user behaviour in terms of the touch location on the screen, the length, direction and duration of a swipe, the velocity and acceleration of a swipe, and the finger pressure and the area covered by a swipe. The extracted features are then used to classify a user using an SVM or kNN classifier. The authors evaluate their approach on a dataset of 41 participants and show that their approach is able to provide an EER of $\leq 3\%$ by using a window size of 13 swipes.

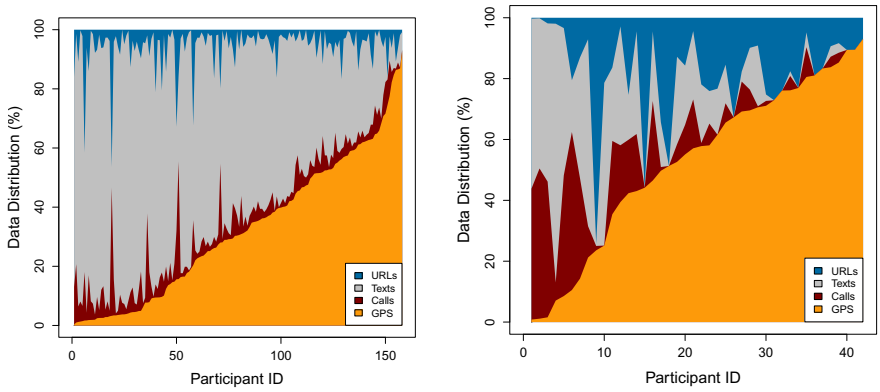
Keystroke Behaviour-Based IA Scheme (Keystroke-IA) [13]. Various classifiers have been proposed that use keystroke behaviour to implicitly authenticate the device owners [8,13,26]. Some keystroke classifiers [8,26] use two features — inter-stroke delay and key holding time (time elapsed between a key press and the corresponding key release event). Furthermore, these classifiers have been tested on multiplexed numeric keypads. In a recent paper [13], the authors employ an additional feature – touch pressure – to provide IA for virtual keypads on modern smartphones. Empirical evaluations on data collected from 40 users and a window size of 15 keystrokes provide an EER of $\leq 10\%$, $\leq 20\%$ and $\leq 5\%$ for J48, Random Forrest and Bayesian classifiers, respectively.

SilentSense [4]. We choose SilentSense as a candidate IA scheme for our evaluations because of the unique feature set that it uses and because of its high detection accuracy ($\sim$ 100%). The authors of SilentSense [4] observe that a combination of the touch input behaviour and the corresponding reaction of a smartphone (micro-movement) can be used to create a more robust model of a user's behaviour. SilentSense operates by combining interacting features from touch behaviour (such as pressure, area, duration, and position) for different touch actions (including fling, scroll, and tap) with the reaction of device features (like acceleration and rotation) to model user behaviour. For the scenarios where the user is walking, the micro-movement patterns are perturbed, since the sensory data generated during walking will skew the sensory data generated due to the reaction of the device to touchscreen interactions. To deal with the walking scenario, the authors extract four features including: (1) vertical displacement of each step; (2) current step frequency; (3) mean horizontal acceleration for each step; and (4) standard deviation of vertical acceleration for each step. They evaluate their approach on a dataset containing data from 10 users and 90 guests. Their evaluations show that by using an SVM classifier, they are able to achieve an EER of $\leq 1\%$ by using a window of three touch strokes.

Context-Aware Touch Behaviour-Based IA Scheme (TIPS) [12]. Feng et al. [12] demonstrate that the EER of a classifier based on touch screen input behaviour reaches up to 40% when it is evaluated on data from multiple applications in an uncontrolled environment. The authors argue that this accuracy degradation is due to variations in usage behaviour. For example, data generated for the same user for different device holding patterns (left hand vs. right hand); for different mobility patterns (stationary vs. walking); and for different applications (maps vs. browser) is different enough to cause accuracy degradation. To mitigate this degradation, the authors propose a multi-stage filtering hierarchy consisting of four levels: (1) foreground application; (2) direction of swipe; (3) swipe length; and (4) swipe curvature. During the one week training period, the prototype of their scheme collected 2000 gestures from 23 smartphone users. After generating the templates by performing multi-stage filtering, the authors were able to achieve an EER of $\leq 10\%$ using a window of eight swipes. Despite the fact that TIPS uses similar features to Touchalytics [15], we choose TIPS for performance evaluation in order to evaluate the impact of intelligent use of contextual information to increase the accuracy of existing IA schemes.

3 Evaluation Datasets

For the empirical evaluation of the IA schemes, we use real-world and unbiased datasets that capture the natural behaviour of the participants. We use two real-world data sets that broadly capture data from devices while users are using them (e.g., location, wireless connections including network, bluetooth and WiFi, contacts, battery status, call logs, text logs, phone orientation, gyroscope and accelerometer readings, and running apps). These datasets are used



(a) Netsense dataset (b) WatApp dataset

Fig. 1. Distribution of URL, text, call and GPS records collected from different participants in the Netsense and WatApp datasets, sorted by fraction of GPS data. Percentages are derived from the number of discrete events collected from each participant.

to evaluate Shi-IA and Gait-IA. However, these datasets do not include touch or keystroke data. We therefore use a third real-world dataset that captures swipe data and use it for evaluating Touchalytics, SilentSense and TIPS. Ideally we would gather day-to-day freehand keyboard input from participants. For privacy reasons, however, we cannot use a user's real-world communications for keystroke data. We therefore have users type predefined email and SMS strings to evaluate Keystroke-IA. In this section, we provide data collection goals, experimental design, and the process used for collecting the four evaluation datasets.

3.1 Netsense Dataset [37]

University of Notre Dame researchers created the Netsense dataset by providing 200 first-year Notre Dame students with Android smartphones. These devices were modified to log many events including contacts, texts, voice calls, Wi-Fi scanning results and current access point, Bluetooth scanning results and connections, browser history, running apps, battery status, location, email, and port traffic. While the purpose of their study was to understand social ties, many of these features overlap with the features used by researchers for IA [35].

Data Statistics. We contacted Striegel et al. [37] to request a chunk of their dataset. They provided us with data that they logged between 2012-11-01 09:34:35 and 2012-11-30 12:49:50. This chunk of the dataset contained data belonging to 158 participants. For our study, we extract the location, call history, text history and browser history data. For these users, we extract 125846, 15003, 244627 and 4817 location events, call events, text events and webpage access events, respectively. The data distribution across participants is plotted in Fig. 1(a). We note that this dataset is not labeled (i.e., there is no way to label the data for instances

when the device was voluntarily given to someone for use by the owner or when it was deliberately misused by a non-owner).

3.2 WatApp Dataset

While the Netsense dataset is useful for our study, we also want to collect labeled data. Therefore, we instrument WatApp[2] (an Android App widely used by University of Waterloo students to get information about current weather, searchable maps, class schedules, and events) to log events on participants' devices. In addition to logging the same data as Netsense, WatApp logs gyroscope readings and accelerometer readings. The sensitive fields are one-way hashed to preserve the privacy of participants. Furthermore, to establish the ground truth, we ask participants to label the intervals for which they are *absolutely certain* that the device was in their possession.

To advertise for participants, we used our university-wide mailing list to advertise for people who would be interested in a study on"Mobile Misuse Detection". Participants were expected to install WatApp on their smartphones for ten weeks. Participants had the option to opt-out any time they wanted by disabling the data collection mode. Furthermore, if they wanted WatApp to not log data, they were provided with the option to pause data collection for an indefinite amount of time. We paid the participants $5 for each week of participation (up to $50 in total for ten weeks of participation).

Data Statistics. Our application was downloaded and installed by 74 participants and 42 of those participants completed the study. In total, we logged 1371908 events over ten weeks. For 42 users, we extracted 121525, 15962, 28958 and 36178 location events, call events, text events and webpage access events, respectively. Data distribution across participants is plotted in Fig. 1(b).

3.3 Touchscreen Input Dataset

Our goal is to collect a dataset that captures the natural behaviour of the participants when they use the touchscreens of their smartphones. We do not want the participants to perform predefined tasks. We also want to study touchscreen input behaviour across a diverse set of applications. Therefore, to capture data that satisfies our data collection goals, we instrument four Android apps: a browser app[3], a maps/ navigation app[4], a launcher app[5] and a comic viewer app[6]. The apps that we choose belong to diverse categories and help us in understanding user behaviour across different apps. To advertise for participants, we used our university-wide mailing list for people who would be interested in a study on smartphones apps. Participants were expected to install these apps on their

[2] `http://play.google.com/store/apps/details?id=watapp.main`
[3] `http://code.google.com/p/zirco-browser/`
[4] `http://code.google.com/p/osmand/`
[5] `http://code.google.com/p/android-launcher-plus/`
[6] `http://code.google.com/p/andcomics/`

Table 1. Statistics of touch points dataset

App.	Num. of touchpoints	Num. of Swipes	Sessions	Mean (Median) swipes per session
Launcher	642442	19740	4417	4.46 (2)
Browser	1164011	20139	826	24.3 (16)
Maps	236878	4664	365	12.7 (8)
Comics	445538	8928	272	32.8 (16)
Total	**2488869**	**53471**	**5880**	**9.09**

smartphones for ten weeks. We did not ask the participants to explicitly perform any tasks and participants were to use these apps as per their needs. This allowed us to capture participants' *in the wild* behaviour. We paid the participants $5 for each week of participation (up to $50 in total for ten weeks of participation).

For data collection, every time a participant interacts with the touchscreen on one of the provided applications, we record: 1) time stamp in milliseconds; 2) x and y co-ordinates of the touch point, 3) finger pressure on the screen; 4) area covered by the finger on the screen; 5) values from the accelerometer sensor; 6) finger orientation; 7) screen's orientation; 8) smartphone's orientation sensor's value (roll, pitch and azimuth); and 9) accelerometer sensor values. These values are temporarily stored on the participant's device and then batch transmitted to a server. Before every data transmission, we establish the ground truth (only the participant used the applications) by asking the participants to label the intervals for which they are *absolutely certain* that the device was in their possession.

Data Statistics. Our applications were downloaded and used by 61 participants. In total, we logged about 2.49 million touch points comprising over 53,000 swipes in ten weeks. The details of swipes, their distribution across applications and distribution across user sessions is provided in Table 1.

3.4 Keystroke Dataset

We want to collect keystrokes of participants during their normal usage sessions; however, this is difficult in a privacy preserving manner. Therefore we present users with text strings that are used in everyday communication. To this end, we choose text strings from existing publicly available SMS [6] and email corpora [23]. We develop an Android app that presents a participant with each string of data that they are expected to input using the virtual keypad on their smartphone. Once a user inputs all the strings, the logged keystroke data is transmitted to our server. To advertise for participants, we used our university-wide mailing list for people who would be interested in a study on "The need for Auto-complete and Auto-correct on Smartphones". To avoid any bias, we do not tell participants about the real purpose of this study before the conclusion of the study. Finally, we do not restrict the participants to complete the study in a limited number of sessions nor ask them to complete it in a lab. We paid $10 to each participant for completing this study.

Data Statistics. We presented participants with 13 strings. These strings contained 43 words and 268 characters in total. We required every participant to input each string four times to collect 1072 keystrokes from each participant. Our application was installed and used by 40 participants. The mean time taken to complete the study was eight minutes.

4 Comparative Evaluation

In this section, we first discuss our experimental setup and then provide the results of our evaluations.

4.1 Evaluation Setup

While most of the evaluation metrics that we use are independent of the underlying implementation language, we wish to measure processing complexity on real Android devices. By using Java as our implementation platform, we are able to measure these statistics easily. Therefore, despite the availability of Matlab source code for Touchalytics [15], we re-implement it in Java.We re-use the publicly available C++ implementation of Gait-IA [14] via the Android Native Development Kit. We note that evaluating the Gait-IA scheme as a native app will result in relatively better results for processing overhead metrics. For the evaluation of other metrics, we used automated scripts on a desktop machine

For our evaluations, we use the recommended parameter values of IA schemes from their original papers. If a paper does not specify a recommended value (e.g., the decay parameter for Shi-IA), we first evaluate the proposed scheme while keeping the classifier threshold to a constant value to determine the best operating point of the tuning parameter for which a recommended value is not provided. To evaluate Shi-IA, we use the Netsense and WatApp datasets. For Gait-IA, we use sensor readings from the WatApp dataset. Keystroke-IA uses the Keystroke dataset for training and classification purposes. Finally, the Touchalytics, SilentSense and TIPS schemes all use the Touchscreen Input dataset.

We construct non-overlapping training and test sets for each of the participants, using negative instances from other users. In practice, it is recommended that IA classifiers come prepackaged with such data to be used as negative instances, allowing robust classifiers to be trained on-device. In our work, the negative training sets of a user for the Keystroke and Touch datasets are constructed by employing usage data from 20 other users. For the Netsense and WatApp datasets, we use one day of data from 14 other users to construct two weeks of negative test data. Frank et al. [14] recommend using a continuous block for training their classifier; consequently, we employ the largest single block of continuous data for training. For Touchalytics, Keystroke-IA, and SilentSense, we use half of the data for training, and the remaining data for testing. In the case of TIPS, we use a 30/70 ratio for training and testing, respectively. This variation in partition ratios is due to us following the convention established in the respective original papers, and due to the heterogeneity of the different types of data used by the different schemes in this work.

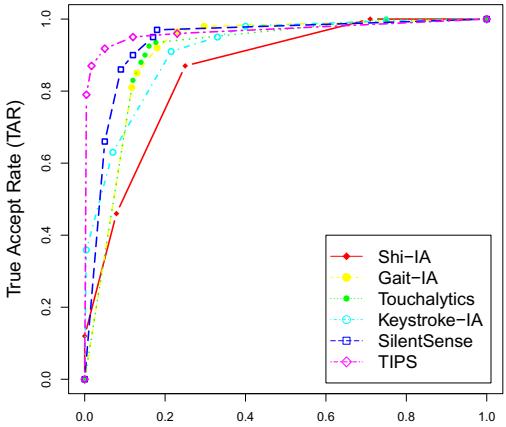


Fig. 2. Accuracy evaluation of the six IA schemes evaluated in this work

4.2 Evaluation Results

Accuracy Evaluation. The accuracy of an IA scheme is its most critical evaluation metric. Ideally, the scheme should have no false rejects (for a seamless user experience of the device owner) and 100% true reject rate (to detect and prevent misuse by an adversary). To understand the accuracy of these classifiers, we plot the ROC curve using the True Accept Rate (TAR) and the False Accept Rate (FAR). To understand the trade-off between TAR and FAR, we threshold the authentication score. Thresholding of Shi-IA is performed over the computed authentication score. Gait-IA and Touchalytics, which use ANN [2] and k-NN for classification, are thresholded over the distance function score and over k, respectively. Keystroke-IA implementation uses a Bayesian Network classifier [16] and is thresholded over the p score. Our implementation of SilentSense uses LIBSVM [5] with a gaussian radial-basis function (rbf) as kernel. For thresholding, we tune the γ and C parameters to rbf. TIPS uses Dynamic Time Warping [3] to compute a similarity score and we threshold the similarity score. The results of the accuracy evaluation, averaged across all users, for the six classifiers are provided in Fig. 2.

As shown, the TIPS scheme outperforms the others in almost all cases. In particular, it is able to achieve a TAR of 79% with a FAR of only 0.43%. TIPS and SilentSense together Pareto dominate all other schemes when the FAR is under 25%. Shi-IA generally underperforms the other schemes, although it has the distinction of being the only IA scheme to achieve a TAR of 100% with a FAR of less then 100% (specifically, 71%). Empirically, this may be due to the fact that Shi-IA uses location information as a discriminator, while the datasets are mostly taken from students living in tightly grouped geographic areas. Consequently, these results may be different for other types of users (e.g., people who travel often). This phenomenon is discussed further in Section 5.

Data Availability. If an IA scheme employs data from a behavioural source that does not have enough data available to make a classification decision for a significant number of usage sessions, the IA scheme would be ineffective despite its high detection accuracy. For example, while Gait-IA outperforms Keystroke-IA in terms of accuracy (see Fig. 2), Gait-IA will not be useful if the device user is stationary and is not generating enough data for classification purposes. We leverage our real-world traces to determine the availability of data for these IA schemes. To compute the data availability we assume that IA is to be performed only once during a session (and not performed repeatedly after a predefined interval of time). We note that an IA scheme may save past authentication scores and re-use them in case data is unavailable (e.g., Gait-IA may compute authentication score prior to the device usage when accelerometer data is available and then reuse this score to authenticate future sessions). However, for a fair comparison, to compute the data availability we only consider data that has been generated during a device usage session.

From the Netsense and WatApp datasets, we calculate the total number of usage sessions (delimited by screen-on events) and the sessions in which enough behavioural features are available to perform a classification decision for Shi-IA, Gait-IA and Keystroke-IA. For keystroke availability, exact keystroke data is not available and so we assume enough data is available whenever the keyboard is displayed on the screen during the session; note that this will lead to some overreporting of keystroke data availability for insufficient data. Since the Netsense and Watapp datasets do not log touchscreen interactions, for Touchalytics, SilentSense and TIPS, we report data availability against the four apps used in the touchscreen input dataset. This will also result in some overreporting of data availability; however, since touchscreen interaction is the primary input mechanism on modern devices, we expect our results to hold for other apps.

As seen in Fig. 3, data derived from touchscreen interaction is almost always available, so IA schemes making use of it are thus most likely to be usable. SilentSense additionally makes use of accelerometer data; when the device is resting on a stable surface this data will not be as meaningful as when the device is being held, but it is still available for training and classification. Availability of data for Shi-IA is highly dependent upon the users' context and is discussed further in Section 5. Gait information was generally the most difficult to find, with enough information available in only 13.1% of sessions.

Training Delay. An IA scheme that could employ data from a few sessions to robustly train itself would be highly desirable. While the IA scheme may explicitly request a user to provide training data (for example, a keystroke classifier asks a user to input a set of strings), most of the existing schemes rely on collecting data during normal usage for training purposes. We utilize the datasets as described in Section 4.1 to determine the training delay for each of the six schemes evaluated in this work. To measure training delay, we set all the tuning parameters including the classification threshold to a constant value and then train the classifier by incrementally increasing the size of training data to the

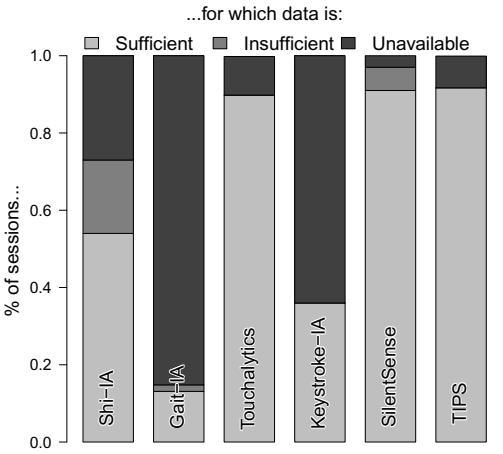


Fig. 3. Data availability on real-world datasets

classifier. For IA schemes that employ classifiers that require negative training instances (e.g., Touchalytics, SilentSense), we use equal amounts of out-of-class training instances from 20 out-of-class sources. For every training session, we measure the accuracy of the classifier by running it on the test dataset. Using this process, we find the minimum number of events and the amount of time required to collect these events to obtain an accuracy of $\geq 70\%$, $\geq 80\%$, and $\geq 90\%$. These results are provided in Table 2.

Training delays are closely correlated with data availability rates. When gait information is available—which is frequently not the case, as discussed previously—Gait-IA takes the least amount of time to accumulate enough information to train a model with high accuracy. Touchalytics and SilentSense take only a few minutes extra, as touch input is a frequent event. Keystroke-IA data takes longer as high accuracy requires the user to type strings that cover a fair amount of the bigram space (as the training data is derived from interstroke timings). The TIPS scheme, despite having the best TAR and FAR overall, requires approximately one hour of data collection to achieve $\geq 90\%$ accuracy. Shi-IA requires several weeks' worth of data, as it relies on user behaviour patterns repeating over large periods of time.

Detection Delay. While the data availability metric determines whether enough data is available across sessions, we evaluate detection delay for these IA schemes to measure the sensitivity of these schemes to misuse attempts. Ideally, we would like the detection delay to be as low as possible to prevent the adversary from accessing confidential data on the device. We measure detection delay in terms of time elapsed from the start of misuse to the time when the IA scheme detects the misuse. For detection delay evaluation, we play back negative instances and look for those that are correctly classified as true rejects by the IA scheme (i.e., we ignore data that results in false accepts).

Table 2. Minimum training delay to achieve accuracy rates of $\geq 70\%$, $\geq 80\%$, $\geq 90\%$. 95% confidence intervals are provided in parentheses. Note that Shi-IA uses the contents of logs as a whole and as such has no concept of an "event".

	Accuracy $\geq$ 70%		**Accuracy $\geq$ 80%**		**Accuracy $\geq$ 90%**	
	Events	**Time (sec)**	**Events**	**Time (sec)**	**Events**	**Time (sec)**
Shi-IA	N/A	1.7 weeks	N/A	3.2 weeks	N/A	N/A
Gait-IA	1434	159 (32)	1832	205 (47)	2338	287 (59)
Touchalytics	67	106 (9.96)	165	280 (30)	275	464 (49)
Keystroke-IA	1352	594 (55)	2028	839 (108)	3380	1101 (360)
SilentSense	86	139 (14)	204	346 (36)	272	460 (49)
TIPS	738	1391 (224)	1295	2443 (378)	1611	3034 (445)

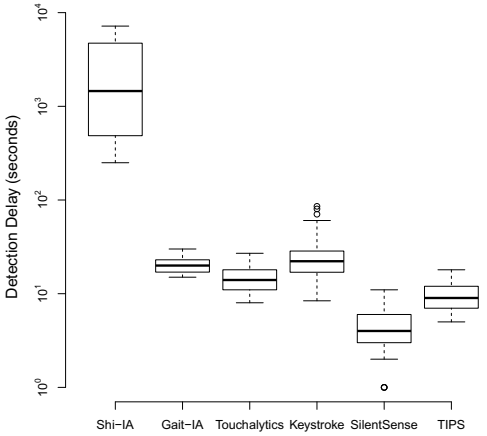


Fig. 4. Detection delay for true rejects (note log scale)

The detection delay results are shown in Fig. 4. SilentSense generally detects non-owners the fastest, in the range of 2-11 seconds. Other schemes generally detect non-owners in less than 30 seconds, with the exception of Shi-IA. Shi-IA takes more than 15 minutes on average before enough data is available for it to reject a non-owner from the device. This result is significantly longer than the average session length, and a malicious user would likely be able to export data from the device before even realizing that an IA scheme is in use.

Processing Complexity. Since the target for these IA schemes is mobile platforms, it is critical for the IA schemes to have low processing complexity. For complexity evaluations, we measure the performance overhead in terms of elapsed CPU time and heap size of the IA scheme for feature collection, training and classification operations. We divide the performance overhead into these operations to distinguish the one-time (training) and run-time (feature collection and

Table 3. Performance evaluation of the IA schemes evaluated in this work. 95% confidence intervals are provided in parentheses. N1: Nexus 1 and N4: Nexus 4.

		CPU (ms)				Heap(kB)
		Init.	**Feat. Extraction**	**Training**	**Classification**	**Runtime**
N1	**Keystroke-IA**	21 (2.08)	<1 (≃0)	<1 (≃0)	0.2 (≃0)	3.2 (0.19)
	Touchalytics	5 (0.27)	0.27 (≃0)	65 (2.16)	1.7 (≃0)	59.1 (1.43)
	SilentSense	1162 (81)	0.75 (≃0)	10384 (91)	0.12 (≃0)	18.3 (1.14)
	Shi-IA	677 (26)	1758 (31)	13053 (87)	58 (4)	790 (6)
	Gait-IA	5 (0.15)	7 (0.24)	764 (42)	93 (7)	9532 (81)
	TIPS	5 (0.18)	0.23 (≃0)	35 (≃1.4)	1.12 (≃0)	92 (2.2)
N4	**Keystroke-IA**	12 (0.95)	<1 (≃0)	<1 (≃0)	0.05 (≃0)	2.9 (0.13)
	Touchalytics	3 (0.27)	0.05 (≃0)	15 (0.5)	1.08 (≃0)	67 (5.59)
	SilentSense	972 (67)	0.55 (≃0)	5937 (329)	0.07 (≃0)	21 (0.68)
	Shi-IA	575 (24)	1406 (22)	10964 (74)	51 (3)	817 (5)
	Gait-IA	4 (0.1)	5 (0.13)	522 (31)	75 (6.8)	9775 (94)
	TIPS	3 (0.18)	0.03 (≃0)	8.2 (≃0.86)	0.73 (≃0)	96.4 (2.5)

classification) costs. An efficient IA scheme would have a reasonable one-time cost and minimal run-time cost.

For execution time calculation, we choose an HTC Nexus 1 and an LG Nexus 4. The Nexus 1 has Android OS v2.1 on a 1GHz processor with 512MB of RAM. The Nexus 4 has Android OS v4.2 on a Quad-core 1.5GHz processor with 2GB of RAM. Execution time results for both devices are provided in Table 3.

The Nexus 4 generally performs operations faster than the Nexus 1, but with marginally higher memory overhead. In our experience, these small differences are generally due to changes in the Android API. SilentSense initialization and training take several seconds due to the SVM classifier used; it also loads negative instances from disk at initializaton. Shi-IA takes 1-2 seconds to extract features from data as it must make a GPS request and also filter call, SMS, and browser logs. All schemes are able to perform classification in tens of milliseconds in the worst case.

Uniqueness of Behavioural Features. Jain et al. [19] list distinctiveness as one of the key properties of a biometric-based authentication system, which requires any two persons to be sufficiently different in terms of the characteristics measured. While the presence of false accepts in Fig. 2 indicates that none of the behavioural features employed in the IA schemes evaluated in this work are distinct, nevertheless they should provide sufficient discriminatory information among a *sufficiently* large set of users to provide an acceptable FAR. To gain insight into this, we simulate N non-owners attempting to access a protected device, and measure the rate at which someone is able to successfully bypass IA. By varying the number N, we gain some sense of the device owner's uniqueness

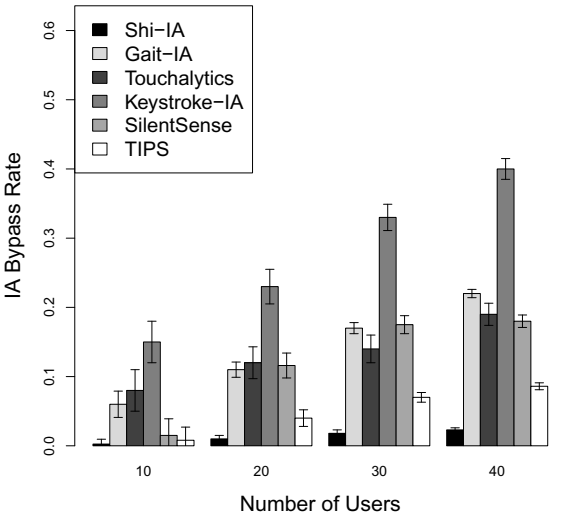


Fig. 5. Relationship between IA bypass rate and number of users

in a crowd of that size. For each value of N, this simulation is run using 4-fold cross-validation for each user and the results are averaged.

Fig. 5 shows the results from this simulation. All of the IA schemes tested appear to exhibit similar growth patterns in IA bypass rate as the number of users increases. While TIPS and Shi-IA exhibit the most uniqueness overall, SilentSense is also quite resilient when faced with 10 or fewer adversaries. Keystroke-IA does not appear to be distinctive even in scenarios with few non-owners present, suggesting that it would be wise to pair these features with other, non-keystroke-derived attributes when creating IA schemes.

Vulnerability to Mimicry Attacks. While a detailed analysis of vulnerability to mimicry attacks is beyond the scope of this paper, in this section we consider the informed adversary threat scenario. An uninformed adversary may be a curious stranger/thief who found/stole a device, while an informed adversary might be an inquisitive friend, co-worker, or family member. The difference between these two types of adversary is that the latter may have additional knowledge about the behaviour of the victim (for example, he may know that the victim always uses his right hand for swiping). Based on the informed adversary scenario, we consider how effortlessly such an adversary can defeat an IA scheme. Interested readers are referred to [31,32] on advanced automated mimicry attack scenarios for touch- and keystroke-based IA schemes.

We argue that in accordance with Kerckhoffs's principle, the IA mechanism (including its features and computation of anomaly score) is public knowledge but feature values for individual users are secret. Consequently, if an adversary can estimate the feature values for an IA scheme easily and mimic those feature values, he can steal data from the device. From the approaches that we evaluate,

Shi-IA is the most vulnerable to mimicry attacks. Even an uninformed adversary can scan the device for call/text logs and browser history and then mimic it to ensure that the device does not lock him out. An informed adversary would attempt to stay in the same vicinity as the device owner to get an even better authentication score.

Other IA schemes evaluated in this work are more difficult to mimic. Some of the schemes rely on features that may be estimated by an informed adversary. For example, in Touchalytics, an adversary may be able to approximate the start and end co-ordinates for swipes. Similarly, for SilentSense, the adversary may be able to coarsely estimate the amount of action force by looking at the reaction of the device. While the aforementioned features for their respective IA schemes are relatively easy to estimate by an informed adversary, most of the features used by these schemes are hidden (not measurable by a naked eye). For example, the touch width of a swipe is hidden to a surveilling adversary. Similarly, the key release time for keystroke classifier are difficult to approximate without special equipment.

A more serious attack surface for these IA schemes exists in that many of the features employed by these schemes can be collected by any app without requiring any special Android permissions (except Shi-IA, which requires the permissions mentioned in § 4.2). Consequently, an adversary might persuade the victim to install a Trojan app on his device in order to log his behaviour. The adversary can then train himself to mimic the victim. Tey et al. [39] mounted this attack on a keystroke-based authentication scheme for traditional keyboards. They demonstrated that by using a carefully designed user interface, they were able to train participants of their study to achieve an average FAR of 63%. It is possible that similar active attacks could be mounted on touch-, keystroke- and gait-based IA schemes, which is an area that needs further study.

Ease of Deployment on Mobile Platform. Finally, we look at the deployment related issues for these IA schemes on the popular iOS and Android platforms. We understand that sufficient changes might be introduced by the OS providers in future versions to mitigate the deployment limitations of these IA schemes; nevertheless, we provide an overview of the deployment issues on contemporary mobile platforms.

The features used by Gait-IA can be collected without requiring any permissions. Features employed by Shi-IA can be collected using non-root permissions. More specifically, on Android five permissions including `ACCESS_FINE_LOCATION`, `READ_SMS`, `READ_CALL_LOG`, `READ_HISTORY_BOOKMARKS` and `READ_CONTACTS` can be used to implement Shi-IA. Feature extraction for touch- and keystroke-based classifiers is more complicated. Due to security and privacy concerns, iOS and Android only allows a foreground app to receive input events (touch and keystroke events). Therefore, IA schemes that employ these features including [4,12,13,15] can only be deployed either on rooted devices or deployed per app instance [22].

5 Discussion and Open Challenges

This section discusses guidelines for creating implicit authentication schemes that we derive from the data collection process, the results in Section 4, and our experience in implementing the schemes on Android devices.

Practical Implicit Authentication Is Possible with Low Overhead and in Near-real-time. Our results on Nexus 1 and Nexus 4 devices given in Table 3 show there are IA schemes that can run feature extraction and classification in only milliseconds. Even the worst case training scenarios take only ten seconds, which is performed one-time only and can be done in a background thread. In terms of accuracy, touch behaviour-based approaches provide $\geq 90\%$ true accepts with $\leq 10\%$ false accepts and are a good candidate for secondary authentication. Finally, in case of misuse by non-owner, the majority of these implicit authentication schemes are able to detect misuse in under 30 seconds.

Features Should be Chosen in a Complementary and Context-aware Manner. Sources for behavioural features must be chosen carefully, and take the intended deployment context into account. Touch-based data is almost always available (Fig. 3) but should be augmented with a secondary source (such as keystrokes) for better coverage. Taking into account user context information – e.g. whether the user is walking or stationary, which app the user is interacting with – is important for classifying data from onboard sensors (TIPS), but does not necessarily make a good discriminator by itself (Shi-IA). No individual source of behavioural data provides a silver bullet for IA.

Devices May Not Need to Be Rooted to Make Use of IA. Android does not allow background applications to gather input events (touch and key input events) due to security concerns. Therefore, IA schemes that rely on input events (e.g. touch- and keystroke-based schemes) require root privileges on the device in order to collect data. On the other hand, Shi-IA and Gait-IA do not require root privileges and only require Android permissions. Input event data can be collected by individual apps without any additional permissions, which opens the door for IA protection at the app level instead of at the device level [22]. For example, enterprises can bundle IA schemes within their apps to protect confidentiality of their corporate data. While providing IA at the app level mitigates the restrictions imposed by Android, it also imposes significant development overhead. All of these are open questions that should be considered when proposing any new IA scheme.

Using a Realistic Threat Model and Evaluating in an Uncontrolled Environment Is Necessary When Evaluating an IA Scheme. Some IA proposals are accompanied by unrealistic evaluations, by having users perform a repeated task in a lab setting to generate data. When these schemes are then applied in real-world settings, the assumptions made in the lab may prove false and the scheme's performance will suffer accordingly. Feng et al. [12] demonstrate that on real-world datasets, many existing touch-based IA schemes have significantly higher EER than reported in the original papers. Our findings are similar for the IA schemes that had their datasets publicly available [14,15].

Furthermore, a recent Symantec study finds that 68% of non-owners who attempted to access private data on an unguarded smartphone did so on the spot, which would make location filtering an unhelpful IA feature [41]. A similar study by Lookout-Sprint [25] found that 44% of users were primarily concerned with their devices being accessed by family and friends, as opposed to strangers. Since such adversaries may have multiple overlapping features (e.g., location and contacts), IA schemes that rely in such features will not be very effective. Therefore, it is critical to provide protection against a realistic threat model that captures these security and privacy concerns of smartphone users.

Mimicry Attacks On IA Schemes Are Possible. In addition to the natural collisions of behaviour we showed in Fig. 5, some researchers have shown deliberately trained attacks on swipes and keystroke input [31,32]. We argue that implicit authentication (i) should be used as a secondary authentication mechanism complementing primary authentication mechanisms, and (ii) should use behavioural features from multiple sources. Using multiple types of characteristics greatly increases the difficulty of building devices that mimic natural human behaviour, and adds dimensions to the complexity of training users to fool behavioural models [36].

6 Conclusion

In this paper we provided a comparative evaluation of six IA schemes that employ different behavioural features. Our empirical evaluations show that IA can be performed with reasonable accuracy and low complexity with acceptable detection delay on contemporary mobile devices. More specifically, our evaluations show that in addition to adequate data availability for training and classification, touch behaviour-based IA schemes outperform other schemes in terms of accuracy and detection delay. We also analyzed real-world traces to show that while keystroke- and gait-based IA schemes provide reasonable performance, there was not enough data available for a significant proportion of sessions to make a classification decision. In terms of evaluation of IA schemes by the research community, our findings emphasize the need for evaluation on uncontrolled datasets and a more realistic threat model. We have made our implementations publicly available to further research in the IA domain.

Acknowledgements. We thank Tao Wang, Sarah Harvey, the anonymous reviewers and our shepherd, Alina Oprea, for their helpful comments. We thank Aaron Striegel, Mario Frank, Jordan Frank and Shu Liu for providing sourcecode of classifiers and datasets. We also thank Google and NSERC for their support.

References

1. Android Authority: Android face unlock hacked (March 2014), `http://androidauthority.com/android-jelly-bean-face-unlock-blink-hacking-105556/`

2. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM (JACM) 45(6) (1998)
3. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD Workshop, vol. 10 (1994)
4. Bo, C., Zhang, L., Li, X.Y., Huang, Q., Wang, Y.: Silentsense: silent user identification via touch and movement behavioral biometrics. In: MobiCom. ACM (2013)
5. Chang, C.C., Lin, C.J.: Libsvm: A library for support vector machines. ACM TIST 2(3) (2011)
6. Chen, T., Kan, M.-Y.: Creating a live, public short message service corpus: The nus sms corpus. Language Resources and Evaluation 47(2), 299–335 (2013)
7. Clarke, N., Karatzouni, S., Furnell, S.: Flexible and transparent user authentication for mobile devices. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 1–12. Springer, Heidelberg (2009)
8. Clarke, N.L., Furnell, S.: Authenticating mobile phone users using keystroke analysis. International Journal of Information Security 6(1) (2007)
9. Crawford, H., Renaud, K., Storer, T.: A framework for continuous, transparent mobile device authentication. Elsevier Computers & Security 39 (2013)
10. De Luca, A., Hang, A., Brudy, F., Lindner, C., Hussmann, H.: Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In: CHI. ACM (2012)
11. Feng, T., Liu, Z., Kwon, K.A., Shi, W., Carbunar, B., Jiang, Y., Nguyen, N.: Continuous mobile authentication using touchscreen gestures. In: HST. IEEE (2012)
12. Feng, T., Yang, J., Yan, Z., Tapia, E.M., Shi, W.: Tips: Context-aware implicit user identification using touch screen in uncontrolled environments. In: HotMobile. ACM (2014)
13. Feng, T., Zhao, X., Carbunar, B., Shi, W.: Continuous mobile authentication using virtual key typing biometrics. In: TrustCom. IEEE (2013)
14. Frank, J., Mannor, S., Precup, D.: Activity and gait recognition with time-delay embeddings. In: AAAI (2010)
15. Frank, M., Biedert, R., Ma, E., Martinovic, I., Song, D.: Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. IEEE TIFS 8(1) (2013)
16. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29(2-3) (1997)
17. Gafurov, D., Helkala, K., Søndrol, T.: Biometric gait authentication using accelerometer sensor. Journal of Computers 1(7) (2006)
18. Hayashi, E., Riva, O., Strauss, K., Brush, A., Schechter, S.: Goldilocks and the two mobile devices: Going beyond all-or-nothing access to a device's applications. In: SOUPS. ACM (2012)
19. Jain, A.K., Ross, A., Prabhakar, S.: An introduction to biometric recognition. IEEE Transactions on Circuits and Systems for Video Technology 14(1) (2004)
20. Jolliffe, I.: Principal component analysis. Wiley Online Library (2005)
21. Kalamandeen, A., Scannell, A., de Lara, E., Sheth, A., LaMarca, A.: Ensemble: Cooperative proximity-based authentication. In: MobiSys. ACM (2010)
22. Khan, H., Hengartner, U.: Towards application-centric implicit authentication on smartphones. In: HotMobile. ACM (2014)
23. Klimt, B., Yang, Y.: Introducing the enron corpus. In: CEAS (2004)
24. Li, L., Zhao, X., Xue, G.: Unobservable reauthentication for smart phones. In: NDSS (2013)

25. Lookout Blog: Sprint-lookout mobile behavior survey (March 2014), http://blog.lookout.com/blog/2013/10/21
26. Maiorana, E., Campisi, P., González-Carballo, N., Neri, A.: Keystroke dynamics authentication for mobile phones. In: SAC. ACM (2011)
27. Mantyjarvi, J., Lindholm, M., Vildjiounaite, E., Makela, S.M., Ailisto, H.: Identifying users of portable devices from gait pattern with accelerometers. In: ICASSP 2005. IEEE (2005)
28. Muaaz, M., Mayrhofer, R.: An analysis of different approaches to gait recognition using cell phone based accelerometers. In: MoMM. ACM (2013)
29. Riva, O., Qin, C., Strauss, K., Lymberopoulos, D.: Progressive authentication: deciding when to authenticate on mobile phones. In: USENIX Security (2012)
30. Schneier on Security: Apple iphone fingerprint reader hacked (March 2014), http://schneier.com/blog/archives/2013/09/apples_iphone_f.html
31. Serwadda, A., Phoha, V.V.: Examining a large keystroke biometrics dataset for statistical-attack openings. ACM TISSEC 16(2) (2013)
32. Serwadda, A., Phoha, V.V.: When kids' toys breach mobile phone security. In: CCS. ACM (2013)
33. Serwadda, A., Phoha, V.V., Wang, Z.: Which verifiers work?: A benchmark evaluation of touch-based authentication algorithms. In: BTAS. IEEE (2013)
34. Shahzad, M., Liu, A.X., Samuel, A.: Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it. In: MobiCom. ACM (2013)
35. Shi, E., Niu, Y., Jakobsson, M., Chow, R.: Implicit authentication through learning user behavior. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 99–113. Springer, Heidelberg (2011)
36. Shrestha, B., Saxena, N., Truong, H.T.T., Asokan, N.: Drone to the rescue: Relay-resilient authentication using ambient multi-sensing. In: Financial Cryptography and Data Security (2014)
37. Striegel, A., Liu, S., Meng, L., Poellabauer, C., Hachen, D., Lizardo, O.: Lessons learned from the netsense smartphone study. In: HotPlanet. ACM (2013)
38. Studer, A., Perrig, A.: Mobile user location-specific encryption (mule): Using your office as your password. In: Wi'Sec. ACM (2010)
39. Tey, C.M., Gupta, P., Gao, D.: I can be you: Questioning the use of keystroke dynamics as biometrics. In: NDSS (2013)
40. Threatpost: Samsung android lockscreen bypass (March 2014), http://threatpost.com/lock-screen-bypass-flaw-found-samsung-androids-030413/77580
41. Wright, S.: Symantec honey stick project. Symantec Corporation (March 2012)
42. Zhao, X., Feng, T., Shi, W.: Continuous mobile authentication using a novel graphic touch gesture feature. In: BTAS. IEEE (2013)